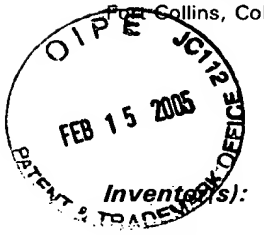
*HEWLETT-PACKARD COMPANY*
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

*PATENT APPLICATION*

*ATTORNEY DOCKET NO.* 10001025-1




***IN THE***
***UNITED STATES PATENT AND TRADEMARK OFFICE***

***Inventor(s):*** William G. Hooper III

***Confirmation No.:*** 1710

***Application No.:*** 09/746,328

***Examiner:*** Hetul B. Patel

***Filing Date:*** 12/20/2000

***Group Art Unit:*** 2186

***Title:*** METHOD AND SYSTEM FOR DATA BLOCK SPARING IN A SOLID-STATE STORAGE DEVICE

***Mail Stop Appeal Brief-Patents***
***Commissioner For Patents***
***PO Box 1450***
***Alexandria, VA 22313-1450***

***TRANSMITTAL OF APPEAL BRIEF***

***Sir:***

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on ***11/10/2004***.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

***(complete (a) or (b) as applicable)***

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

(X) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

| | | |
|---|---|---|
| (X) | one month | $120.00 |
| ( ) | two months | $450.00 |
| ( ) | three months | $1020.00 |
| ( ) | four months | $1590.00 |

02/16/2005 MAHMED1 00000017 082025 09746328
02 FC:1251 120.00 DA

( ) The extension fee has already been filled in this application.

( ) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account ***08-2025*** the sum of ***$620.00***. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: ***Feb. 10, 2005***

***OR***

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number_____ on _____

Number of pages:

Typed Name: ***Joanne Bourquignon***

Signature: Joanne Bourquignon

Respectfully submitted,

***William G. Hooper III***

By Robert W. Bergstrom

***Robert W. Bergstrom***

Attorney/Agent for Applicant(s)
Reg. No. ***39,906***

Date: ***Feb. 10, 2005***

Telephone No.: ***206.621.1933***

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re patent application of:

Inventor: William G. Hooper III
Serial No. 09/746,328
Filed: December 20, 2000
For: Method and system for data block sparing in a solid-state storage device

Examiner: Hetul B. Patel
Group Art Unit: 2186
Docket No. 10001025-1
Date: February 10, 2005

---

## APPEAL BRIEF

Commissioner of Patents and Trademarks
Washington, DC 20231

Sir:

This appeal is from the decision of the Examiner, in an Office Action mailed on August 10, 2004, finally rejecting claims 1-13, 15 and 17.

## REAL PARTY IN INTEREST

The real party in interest is Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 20555 S.H. 249 Houston, TX 77070, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

## RELATED APPEALS AND INTERFERENCES

Applicant's representative has not identified, and does not know of, any other appeals of interferences which will directly affect or be directly affected by or have a bearing on the

Board's decision in the pending appeal.

## STATUS OF CLAIMS

Claims 1-18 are pending in the application. Claims were finally rejected in the Office Action dated August 10, 2004. Applicant's appeal the final rejection of claims, which are copied in the attached CLAIMS APPENDIX.

## STATUS OF AMENDMENTS

No Amendment After Final is enclosed with this brief. The last Response was filed June 11, 2004. The last amendment to the claims was filed January 5, 2004.

## SUMMARY OF CLAIMED SUBJECT MATTER

### Overview

Embodiments of the current invention are directed to a method for sparing defective data blocks in a solid-state data-storage device. When a data block is determined to be defective, a spare block can be used to replace the defective data block. Modern data-storage devices often employ logical block addresses for access to data blocks. These modern data-storage devices arrange for the logical block address corresponding to a block of information to remain constant, regardless of whether the physical block address, or physical location, of the physical block storing the block of information changes as a result of the remapping of a defective data block to a spare data block.

One embodiment of the present invention is concisely described, beginning on line 30 of page 5 of the current application, as follows:

> Figure 2 illustrates the high-level organization of electronic memory of a general embodiment of the present invention. The organization as illustrated in Figure 2 is superimposed upon a sequential block organization of memory that is itself superimposed on a linear, sequentially addressed byte organization described in Figure 1. The memory, or memory address space, is divided into five regions: (1) low-address spare tables 201, a region containing one spare table for each data page, each spare table containing one element for each data block within a data page, each element of a spare table containing a spare status and spare offset to indicate whether or not the corresponding block of a corresponding data page has been remapped and, in the case that the data block within the data page has been remapped, the offset used to locate the replacement block within a spare page; (2) low-address spare pages 202, a region containing a spare page for each data page, each spare page containing a fixed number of replacement, or spare, data blocks; (3) data pages 203, a region

containing data pages, each data page containing a fixed number of data blocks; (4) high-address spare pages 204, a region containing a spare page for each page in the data pages region, each spare page containing a fixed number of spare data blocks; and (5) high-address spare tables 205, a region containing redundant copies of the spare tables contained in the low-address spare tables region 201.

Figure 3 illustrates block addressing employed in a generalized embodiment of the present invention. A solid-state data storage device provides a block-read and block-write interface for accessing devices, such as computers. The accessing device specifies a block for reading or writing using a logical data block address ("LDBA") 302. In a preferred embodiment of the present invention, an LDBA contains a total of 32 bits or is, in other words, a 32-bit unsigned integer, but alternative embodiments may use different sized LDBAs. From the standpoint of the accessing device, the accessing device may access any one of $2^x$ data blocks stored within the solid-state data storage device, where $x$ is some number of bits equal to, or less than, the total of bits that compose an LDBA. However, from the standpoint of the solid-state data storage device, an LDBA is composed of two fields: (1) a data-block-index field 303; and (2) a page-index field 304. In a preferred embodiment of the present invention, the data-block-index field is composed of sixteen lower address bits and the page-index field is composed of sixteen higher-address bits within the LDBA.

The solid-state storage device translates an LDBA into a physical address in order to carry out a read-block or write-block operation specified in terms of an LDBA. Figure 4 is a flow-control diagram of this process, and the process will be described with reference to both Figure 3 and to Figure 4. First, the solid-state storage device, in step 401 of Figure 4, resolves a supplied LDBA 302 into a page index 304 and a data-block index 303. Next, in step 402, the solid-state storage device determines whether the spare table corresponding to the page index has been currently loaded into a high-speed local memory. To make this determination, the solid-state storage device compares a register that contains the page index corresponding to the currently loaded spare table to the page index extracted from the LDBA in step 401. If the currently-loaded spare table is not the spare table that corresponds to the page index extracted from the LDBA, then in step 403, the solid-state data storage device uses the extracted page index 304 as an offset into the low-address spare table region 201 to retrieve the spare table 305 corresponding to the extracted page index 304. Should the solid-state device fail to retrieve the spare table from the low-address spare table region, a redundant copy of the spare table may be retrieved instead from the high-address spare table region. Next, in step 404, the solid-state storage device employs the data block index extracted from the LDBA as an offset into the spare table 305 to select a spare table element 306 corresponding to the LDBA. In step 405, the solid-state storage device uses a data field within the spare table element 306 to determine whether or not the LDBA has been remapped to a spare block. If the LDBA has been remapped, then steps 406-408 are carried out to determine the physical address of the spare block corresponding to the LDBA. Otherwise, steps 409 and 410 are carried out to determine the physical address of the data block within the data page region of memory 303 corresponding to the LDBA. In step 406, the solid-state storage device extracts an offset from the spare table entry 306. In step 407, the solid-state storage device selects a spare page from either the low-address spare page region 202 or the high-address spare page region 204. The status extracted from the spare table element 306 indicates from which spare page

> region to select the spare page. The solid-state storage device uses the extracted page index as an offset into the appropriate spare pages region to select the spare page 307 that contains the spare data block corresponding to the LDBA. Finally, in step 408, the solid-state storage device uses the offset extracted from the spare table element 306 in step 406 as a block offset within the spare page 307 to locate the physical block 308 corresponding to the LDBA. When no remapping of the LDBA has occurred, then in step 409, the solid-state storage device uses the extracted page index 304 as an offset into the data page region 203 to locate the data page 309 corresponding to the LDBA. In step 410, the solid-state storage device uses the data-block index 303 as an offset into the data page 309 to identify the physical data block 310 associated with the LDBA.

Applicant appreciates and understands that bad-block replacement schemes have been employed for magnetic disk drives for many years. Applicant states, in the Background of the Invention section, beginning on line 15 of page 1:

> As with any physical material, the surfaces of rotating magnetic media are subject to manufacturing defects and defects that arise during use due to mechanically and electrically induced stresses. In order to enhance the reliability of magnetic media, sophisticated defect-circumventing mechanisms have been developed to map defective data storage regions of a magnetic medium to available, unused, spare data storage regions provided on the magnetic medium. A variety of methods for remapping defective areas have been developed and are currently in use. Most depend on provision of extensive lookup tables that are interspersed with data-containing regions of the magnetic medium.

As Applicant has carefully pointed out, on page 2 of the specification in the Background of the Invention section, providing spare memory cells for solid-state, electronic memories involves different considerations and constraints than providing bad-block replacement in magnetic disk drives:

> Just as regions of the surfaces of magnet disk drives may contain manufacturing defects, or may become defective through use, data-storage cells within an electronic memory may be defective upon manufacture or may fail during use. Just as in magnetic disk drives, solid-state storage devices need to provide enhanced overall reliability by detecting defective memory cells and providing spare memory cells as substitutes for defective memory cells. However, magnetic data storage medium is relatively cheap, so that use of a relatively large fraction of the physical data storage medium for remapping tables in magnetic disk drives does not significantly increase the overall cost of a magnetic disk drive. *Moreover, because of relatively long latency times for data access, arising from the need to mechanically position read/write heads over a target data storage region, complex remapping calculations may be undertaken in magnetic disk drives without significantly increasing access times and decreasing data transfer rates.* In solid-state storage devices, by contrast, the physical storage medium is expensive, and therefore the use

of a relatively large fraction of the medium for remapping tables can significantly increase the overall price of a solid-state storage device and significantly decrease the solid-state storage device's cost effectiveness in a given application, *and complex remapping calculations directly increase access times and decrease data transfer rates. For these reasons, designers, manufacturers, and users of solid-state storage devices have recognized the need for a method and system for dynamically substituting spare memory cells to replace defective memory cells in the solid-state storage device that does not employ large remapping tables and complex remapping calculations.* (emphasis added)

Independent Claim 1

Claim 1 claims a solid-state storage device comprising: (1) a physical electronic memory including a spare table region (201 in Figure 2) containing spare tables (201 and 205 in Figure 2), a spare page region (202 and 204 in Figure 2) containing spare pages, and a data page region (203 in Figure 2) containing data pages; (2) an electronic memory interface that provides, to devices that access the electronic memory, memory operations directed to target data blocks specified by the accessing device via a logical data block address; and (3) a logic component that maps a logical data block address to a physical address describing the location of a data block in the electronic memory (Figures 3 and 4 and the above quoted text from the current application).

Dependent Claims 2 – 11

Claims 2 – 11 include language directed to additional details about the solid-state storage device claimed in claim 1, including: (1) details regarding the contents of the logical block address, shown in Figure 3 and discussed in the above-quoted text from the current application, claimed in claim 2; (2) the mapping operation carried out by the solid-state logic device, shown in Figure 4, claimed in claim 3; (3) additional details regarding the solid-state-device memory layout, shown in Figure 2, claimed in claims 4-7; (4) additional details concerning the mapping operation carried out by the solid-state logic device, shown in Figure 4, claimed in claim 8; and (5) additional details regarding the solid-state-device memory layout, shown in Figure 2, claimed in claims 9-11.

Independent Claim 12

Claim 12 is directed to one embodiment of a method for transforming a logical data block address into a physical electronic memory address, as shown in Figure 4 and described in the above-quoted text.

Dependent Claims 13 – 18

Claims 13-18 are directed to additional details concerning one embodiment of a method for transforming a logical data block address into a physical electronic memory address, as shown in Figure 4 and described in the above-quoted text.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. The rejection of claims 1-2, 4-5, and 9-12 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck, U.S. Patent No. 6,535,995 ("Dobbeck") in view of Bruce et al., U.S. Patent No. 5,000,006 ("Bruce").

2. The rejection of claims 3, 13, and 17 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck in view of Bruce and further in view of Jeddeloh, U.S. Patent No. 5,933,852 ("Jeddeloh").

3. The rejection of claim 6 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck in view of Bruce and further in view of Venkatesh et al., U.S. Patent No. 6,397,292 ("Venkatesh").

4. The rejection of claims 7-8 and 15 as being unpatentable over Dobbeck in view of Bruce and further in view of Smith, U.S. Patent No. 6,269,432 ("Smith").

ARGUMENT

**ISSUE 1**

1. The rejection of claims 1-2, 4-5, and 9-12 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck in view of Bruce.

Dobbeck discloses a virtual in-line sparing technique for magnetic-disk storage devices. Dobbeck discusses a first prior-art sparing technique, with reference to Figure 1 of Dobbeck, in which physical block addresses ("PBAs"), in column 2, are mapped to logical block addresses ("LBA"), in column 104. As shown in Figure 1, the physical block addresses are sequential, monotonically increasing integers that reference contiguous physical

blocks, while he logical block addresses are not monotonically increasing, since defective or spared physical blocks, shown by the letters "X" and "S," respectively, in the middle column of the table of Figure 1, have no logical block addresses, and are skipped in the logical-block-address sequence. Clearly, this prior-art method does not employ a spare page region containing the spare pages for the device, but instead has spare blocks mixed with data blocks within consecutively addressed physical blocks. There is also no spare page table in this prior-art technique.

Next, Dobbeck discusses a second, prior-art method for disk block sparing, with reference to Figure 2 of Dobbeck, in which a virtual track table 204 and a virtual sector table 206 that together provide an indexing method for translating a logical block address 202 into a physical block address. As stated beginning on line 22 of column 3: "[T]he VT table 204 is indexed by the virtual track, and each entry indicates the cumulative number of skip sectors on the disk prior to that track." As stated beginning on line 27 of column 3:

> Because the VS table 206 contains entries only for the skip sectors and not for all virtual sectors, each VS table entry identifies the virtual sector number of the first good virtual sector in the virtual track identified by the VT table 204 following the skip. That is, the LBA would indicate the PBA, but for the presence of skip sectors. Therefore, it is necessary to add the cumulative number of skip sectors to the LBA to get the next available data sector.

Finally, as stated beginning on line 45 of column 3:

> The calculation of the PBA may be represented by Equation (1) below:
>
> PBA=FLBA+FIND(first S[*p*]>VS, where *p* <= VT[FLBA]<VT[FLBA+1])
>
> where FLBA corresponds to the file LBA, S[*p*] is the skip sector number entry in the VS table 206, VS is the virtual sector number, or the low order bits of the FLBA, and VT is the virtual track number of the current track, so that the FIND function finds the first skip vector number (the index location) in the VS table that is greater than the low order bit value in the LBA, for the current virtual track.

It is manifestly clear that this second prior-art sparing method also mixes spare and defective blocks along with data blocks within one memory block region indexed by successive physical block addresses. Otherwise, the above described calculation cannot work. Thus, this second prior-art technique does not employ a spare page region containing the spare pages for the device, but instead has spare blocks mixed with data blocks within consecutively addressed physical blocks.

Finally, Dobbeck discusses the claimed technique for sparing, based on the

second prior-art technique, in which a prototype table $VS_{pr}$ and a reassign table $VS_{re}$ are added, to decrease the overall sizes of the combined VT, VS, $VS_{pr}$, and $VS_{re}$ tables. The prototype and reassign tables complicates the computation of a physical block address corresponding to a logical block address, but decreases the storage requirements for the table. Thus, Dobbeck's technique, and the second prior-art technique, trade increased complexity of physical block address computation for decreased data-storage-device overhead. This is well summarized in the paragraph beginning on line 8 of column 4:

> Disk defect management using VT and VS tables to perform LBA-to-PBA translation can be implemented relatively easily in software that is stored in program memory of the disk drive and typically is executed by the CPU or state machines of the disk drive formatter. It is anticipated that an increasing amount of defect management software will be located in firmware of the disk drive, because performance speed is increased and demand on the disk drive resources is increased. This will place increased importance on reduced utilization of disk drive resources such as memory.

Thus Dobbeck is quite dissimilar from the currently claimed invention, where, as discussed in the above quoted passage from the current application, both data-storage-device space and computational complexity are minimized. The claimed invention avoids computational complexity partly by avoiding table searches during physical-block-address calculations, as shown in Figures 3 and 4 of the current application, and discussed above in the quoted sections of the current application that refer to Figures 3 and 4. This is possible when the solid-state-storage-device memory is laid out as shown in Figure 2, with separate and discrete spare tables, spare-page regions, and a data page region. As shown in Figures 3 and 4, the page-index portion of a logical block address can be used to index directly into the spare table, spare page region, and data block region, and the data-block-number portion of the LBA can be used to directly index an entry in the spare table and in a data page. There are no table searches needed. By contrast, in Dobbeck, a table search is explicitly included in the formula for computing a physical-block address as the FIND function, and is needed because spare data blocks are intermixed with normal data blocks, and cannot therefore be directly indexed.

In summary, Dobbeck is simply unrelated to the current invention. It is directed to magnetic-disk storage devices, as acknowledged by the Examiner, rather than to solid-state data-storage devices, to which the current invention is directed. More importantly, the current invention uses separate and distinct spare tables, spare page regions, and a normal data page regions, while all sparing techniques mentioned in Dobbeck intermix spare blocks

with data blocks.

Bruce is cited by the Examiner, apparently, for a single sentence in Bruce's abstract: "A flash-memory system provides solid-state mass storage as a replacement for a hard disk." The Examiner apparently feels that the combination of a magnetic-disk-based sparing system completely dissimilar to the claimed solid state storage device with this sentence from Bruce's abstract renders the claimed invention obvious. However, it does not. Although such a combination might make obvious a solid-state data-block sparing system that uses the same layout, tables, and techniques of Dobbeck, the combination is unrelated to the claimed invention.

It should be noted that the block remapping techniques used in Bruce are quite similar to the first prior-art technique discussed in Dobbeck, as can be easily ascertained by comparing Bruce's Figure 3 with Dobbeck's Figure 1.

Please consider claim 1 of the current application:

> 1. A solid-state storage device comprising:
>
> a physical electronic memory including a spare table region containing spare tables, a spare page region containing spare pages, and a data page region containing data pages;
>
> an electronic memory interface that provides, to devices that access the electronic memory, memory operations directed to target data blocks specified by the accessing device via a logical data block address; and
>
> a logic component that maps a logical data block address to a physical address describing the location of a data block in the electronic memory.

In the first element, the solid state storage device is claimed to include a physical electronic memory including a spare table region, a spare page region containing spare pages, and a data page region. Both Bruce and Dobbeck intermix spare blocks with data blocks, and do not employ a spare page region. With regard to establishing a prima facie case for obviousness, as stated in MPEP § 2143, citing *In re Vaeck*, "[T]he prior art reference (or references when combined) must teach or suggest all the claim limitations." Therefore, in rejecting claim 1 based on Dobbeck and Bruce, the Examiner has not established a *prima facie* case for obviousness.

Please next consider claim 2 of the current application:

> 2. (original) The solid-state storage device of claim 1 wherein a logical data block address comprises:
>
> a page index that indexes a data page within the data page region, a spare table within the spare table region, and a spare page within the spare page region; and

> a data block index that indexes a data block within the data page indexed by the page index and a spare table element within the spare table indexed by the page index.

This claim is directed to both the memory layout and indexing of data and spare pages by logical-block-address components. The page index portion of a logical-data-block address indexes a data page within the data page region, a spare table within the spare table region, and a spare page within the spare page region. No portion of a logical-data-block address indexes a spare page region in either Dobbeck or Bruce. As discussed above, because spare blocks are intermixed with data blocks in both Dobbeck and Bruce, neither Dobbeck nor Bruce include spare page regions. Moreover, spare blocks cannot be indexed by any means, in Dobbeck, but instead, must be searched for in a table search.

Claims 4-5 and 9-12 include the limitations of claim 1, include additional limitations, and are therefore, like claim 2, not obvious in view of Dobbeck, Bruce, or Dobbeck and Bruce in combination.

**ISSUE 2**

2. The rejection of claims 3, 13, and 17 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck in view of Bruce and further in view of Jeddeloh.

Jeddeloh discloses a memory with a simple remapping table to direct memory addresses to non-defective memory locations. Jeddeloh, like Bruce and Dobbeck, as discussed with respect to issue 1, neither teaches, suggests, nor mentions the clearly claimed solid-state data storage device of claim 1, including separate spare table, spare page, and data page regions, and Jeddeloh, like Bruce and Dobbeck, neither teaches, suggests, nor mentions the currently claimed method for translating logical block addresses to physical block addresses that uses the spare table, spare page region, and data page region to provide indexing into each table based on a logical block address. No combination of Jeddeloh, Bruce, and Dobbeck makes the currently claimed invention obviousness.

**ISSUE 3**

3. The rejection of claim 6 under 35 U.S.C. § 103(a) as being unpatentable over Dobbeck in view of Bruce and further in view of Venkatesh.

Venkatesh discloses asymmetrical striping of mirrored storage device arrays. Venkatesh is completely unrelated to the claimed invention. Venkatesh, like Bruce and Dobbeck, as discussed with respect to issue 1, neither teaches, suggests, nor mentions the clearly claimed solid-state data storage device of claim 1, including separate spare table, spare page, and data page regions, and Venkatesh, like Bruce and Dobbeck, neither teaches, suggests, nor mentions the currently claimed method for translating logical block addresses to physical block addresses that uses the spare table, spare page region, and data page region to provide indexing into each table based on a logical block address. No combination of Venkatesh, Bruce, and Dobbeck makes the currently claimed invention obviousness.

**ISSUE 4**

3. The rejection of claims 7-8 and 15 as being unpatentable over Dobbeck in view of Bruce and further in view of Smith.

Smith discloses a distributed transactional processing system. Smith is completely unrelated to the claimed invention. Smith, like Bruce and Dobbeck, as discussed with respect to issue 1, neither teaches, suggests, nor mentions the clearly claimed solid-state data storage device of claim 1, including separate spare table, spare page, and data page regions, and Smith, like Bruce and Dobbeck, neither teaches, suggests, nor mentions the currently claimed method for translating logical block addresses to physical block addresses that uses the spare table, spare page region, and data page region to provide indexing into each table based on a logical block address. No combination of Smith, Bruce, and Dobbeck makes the currently claimed invention obviousness.

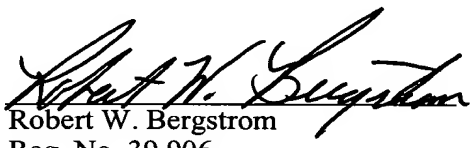
## CONCLUSION

As discussed above, Dobbeck discloses a sparing technique used in disk-based storage devices quite unrelated to the claimed method for sparing data blocks in a solid-state data-storage device. The claimed solid state data storage device uses a data-storage layout that includes a separate spare table, spare page region, and data page region. This enables direct, logical-data-block-based indexing into all three regions, and therefore a rapid and computationally efficient method for accessing a spare block corresponding to logical-data-block address. Because Dobbeck intermixes spare blocks with data blocks throughout the

disk-based data storage device, Dobbeck needs to use computationally inefficient table searches, acceptable, because of the long latency in accessing rotating data-storage media. Dobbeck also seeks to minimize table size at the expense of computational inefficiency. The current method and system seeks to minimize both table space and computational inefficiency. The claimed method and system for sparing blocks in solid state data storage devices are quite dissimilar the sparing methods for disk-based storage devices disclosed by Dobbeck. Bruce uses a straightforward, well-known remapping technique for remapping blocks in a flash memory, unrelated to the currently claimed method and system. Neither Dobbeck, Bruce, or a combination of Dobbeck and Bruce teach, mention, or suggest the currently claimed invention. All of the rejections depend exclusively or primarily on a combination of Dobbeck and Bruce, and therefore all fail for failing to teach, mention, or suggest separate spare page regions within a data storage device, and logical-block-address-based direct indexing into a spare table, spare page region, and data region.

Applicant respectfully submits that all statutory requirements are met and that the present application is allowable over all the references of record. Therefore, Applicant respectfully requests that the present application be passed to issue.

Respectfully submitted,
Robert Alan Cochran
*OLYMPIC PATENT WORKS PLLC*

By 
Robert W. Bergstrom
Reg. No. 39,906

Olympic Patent Works PLLC
P.O. Box 4277
Seattle, WA 98104
206.621.1933 telephone
206.621.5302 fax

## CLAIMS APPENDIX

1. (original) A solid-state storage device comprising:

a physical electronic memory including a spare table region containing spare tables, a spare page region containing spare pages, and a data page region containing data pages;

an electronic memory interface that provides, to devices that access the electronic memory, memory operations directed to target data blocks specified by the accessing device via a logical data block address; and

a logic component that maps a logical data block address to a physical address describing the location of a data block in the electronic memory.

2. (original) The solid-state storage device of claim 1 wherein a logical data block address comprises:

a page index that indexes a data page within the data page region, a spare table within the spare table region, and a spare page within the spare page region; and

a data block index that indexes a data block within the data page indexed by the page index and a spare table element within the spare table indexed by the page index.

3. (previously presented) The solid-state storage device of claim 2 wherein, in order to map a logical data block address to a physical address, the logic component:

extracts the page index and data block index from the logical data block address;

uses the page index to locate a corresponding spare table;

uses the data block index to locate a corresponding spare table element within the corresponding spare table;

when a status indication within the corresponding spare table element indicates that the logical data block address has been remapped,

uses a page offset within the corresponding spare table element and the page index extracted from the logical data block address to determine the physical address of a spare data block, within a spare page, that corresponds to the logical data block address; and

when a status indication within the corresponding spare table element indicates that the logical data block address has not been remapped,

uses the page index and data block index to determine the physical address of a data block, within a data page, that corresponds to the logical data block address.

4. (original) The solid-state storage device of claim 2 wherein the electronic memory includes:

a first spare table region;

a first spare page region;

a data page region;

a second spare page region; and

a second spare table region.

5. (original) The solid-state storage device of claim 4 wherein the first spare table region occupies a first, lower addressed portion of the electronic memory, the first spare page region occupies a second, next lower addressed portion of the electronic memory, the data page region occupies a third, middle portion of the electronic memory, the second spare page region occupies a fourth, higher addressed portion of the electronic memory, and the second spare table region occupies a fifth, highest addressed portion of the electronic memory.

6. (original) The solid-state storage device of claim 4 wherein redundant copies of the spare tables of the first spare table region are stored in the second spare table region.

7. (original) The solid state storage device of claim 4 further including:

a spare table cache and a cached spare table identifier register.

8. (previously presented) The solid state storage device of claim 7 wherein, in order to map a logical data block address to a physical address, the logic component:

extracts the page index and data block index from the logical data block address;

uses the page index to locate a corresponding spare table;

determines, by comparing the page index extracted from logical data block address to the contents of the cached spare table identifier register, whether the spare table cache contains the contents of the located corresponding spare table;

when the spare table cache does not contain the contents of the located corresponding spare table,

copies the located corresponding spare table into the spare table cache;

uses the data block index to locate a corresponding spare table element within the spare table cache;

when a status indication within the corresponding spare table element indicates that the logical data block address has been remapped,

uses a page offset within the corresponding spare table element and the page index extracted from the logical data block address to determine the physical address of a data block, within a spare page, that corresponds to the logical data block address; and

when a status indication within the corresponding spare table element indicates that the logical data block address has not been remapped,

uses the page index and data block index to determine the physical address of a data block, within a data page, that corresponds to the logical data block address.

9. (original) The solid-state storage device of claim 5 wherein a logical data block contains a 16-bit page index and a 16-bit data block index, wherein a data page contains up to $2^{16}$ data blocks, wherein a spare page contains up to $2^7$ - 1 data blocks, wherein the data page region contains up to $2^{16}$ data pages, wherein both the first and second spare page regions contain up to $2^{16}$ spare tables, wherein a spare table includes up to $2^{16}$ spare table elements, and wherein both the first and the second spare table regions contain $2^{16}$ spare tables.

10. (previously presented) The solid-state storage device of claim 1 wherein a spare table contains a number of elements equal to the number of data blocks in a data page, wherein the spare table region contains a number of spare tables equal to the number of data pages within the data page region, wherein the spare page region contains a number of spare pages equal to the number of data pages within the data page region, and wherein a spare page contains a fixed number of data blocks, including a first data block that contains a spare block status map that contains spare block status map elements that store status information for data blocks stored within the spare page.

11. (original) The solid-state storage device of claim 1 wherein a logical data block contains a 16-bit page index and a 16-bit data block index, wherein a data page contains up to $2^{16}$ data

blocks, wherein a spare page contains up to $2^7$ - 1 data blocks, wherein the data page region contains up to $2^{16}$ data pages, wherein a spare page region contains up to $2^{16}$ spare tables, and wherein a spare table includes up to $2^{16}$ spare table elements.

12. (original) A method for transforming a logical data block address into a physical electronic memory address, the method comprising:

providing an electronic memory having a spare table region containing spare tables, a spare page region containing spare pages, and a data page region containing data pages;

extracting a page index and a data block index from the logical data block address; and

using the extracted page index and data block index to identify the address of a physical data block within the electronic memory corresponding to the logical data block address.

13. (original) The method of claim 12 further including:

using the extracted page index as an offset to locate a spare table within the spare table region;

using the extracted data block index as an offset to locate a spare table element within the spare table;

when a status indication within the spare table element indicates that the logical data block address has been remapped,

using the page index extracted from the logical data block address to determine the physical address of a spare page within the spare page region;

using a page offset within the spare table element to determine the physical address of a data block within the spare page; and

when a status indication within the corresponding spare table element indicates that the logical data block address has not been remapped,

using the page index from the logical data block address to determine the physical address of a data page within the data page region, and

using the data block index as an offset to determine the physical address of a data block within the data page.

14. (previously presented) The method of claim 13 further including:

when a status indication within the spare table element indicates that the logical data block address has been remapped,

using the status indication to determine whether the data block is in a low spare page region or a high spare page region and to select the indicated spare page region;

using the page index extracted from the logical data block address to determine the physical address of a spare page within the selected spare page region;

using a page offset within the spare table element to determine the physical address of a spare data block within the spare page.

15. (original) The method of claim 13 further including:

after using the extracted page index as an offset to locate a spare table within the spare table region, and before using the extracted data block index as an offset to locate a spare table element within the spare table,

checking a register to determine whether a currently cached spare table corresponds to the logical block address, and when the currently cached spare table does not correspond to the logical block address, copying the located spare table within the spare table region to a memory cache.

16. (original) The method of claim 15 further including:

when the currently cached spare table does not correspond to the logical block address, and when the contents of the currently cached spare table has been changed since the currently cached spare table was copied to the memory cache, copying the currently cached spare table back to the spare table region prior to copying the located spare table within the spare table region to the memory cache.

17. (original) A method for remapping a logical data block address to a different physical electronic memory address, the method comprising:

providing an electronic memory having a spare table region containing spare tables, a spare page region containing spare pages, and a data page region containing data pages;

extracting a page index and a data block index from the logical data block address;

using the extracted page index and data block index to select the address of a physical data block within a spare page in the spare page region of the electronic memory to which to remap the logical data block address; and

storing indications within an element of a spare table corresponding to the logical data block address that remap the logical data block address to the selected address.

18. (previously presented) The method of claim 17 further including:

using the extracted page index as an offset to locate a spare table within the spare table region;

using the extracted data block index as an offset to locate a spare table element within the spare table;

when a status indication within the spare table element indicates that the logical data block address has been remapped,

using the page index extracted from the logical data block address to determine the physical address of a spare page within the spare page region;

using a page offset within the spare table element to determine the physical address of a spare block status map element within a spare block status map in the first data block within the spare page corresponding to the logical data block address;

setting an indication in the spare block status map element to indicate that a corresponding data block within the spare page is no longer used;

searching the spare block status map within the spare block status map in the first data block within the spare page to select a spare block status map element indicating that a corresponding data block is available;

setting an indication in the selected spare block status map element indicating that the corresponding data block is in use; and

setting indications in the spare table element to indicate that the logical data block address is remapped to data block within the spare page corresponding to the selected spare block status map.

## EVIDENCE APPENDIX

None.

## RELATED PROCEEDINGS APPENDIX

None.